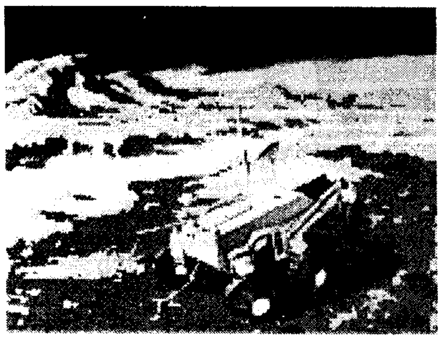
Remote Exploration and Experimentation Project

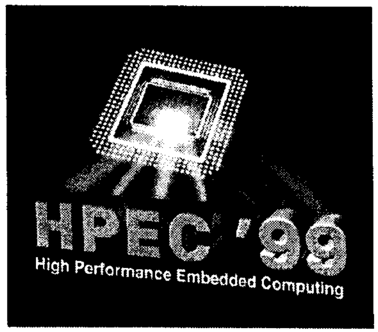
Applications Development for a Parallel COTS Spaceborne Computer

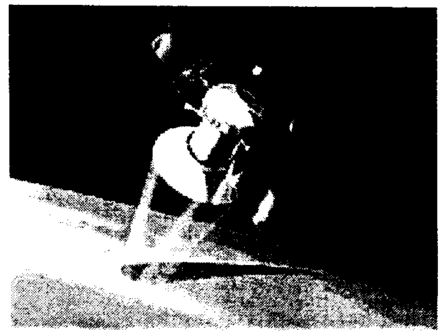
Daniel S. Katz, Paul L. Springer, Robert Granat, and Michael Turmon

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California



Autonomous Vehicles



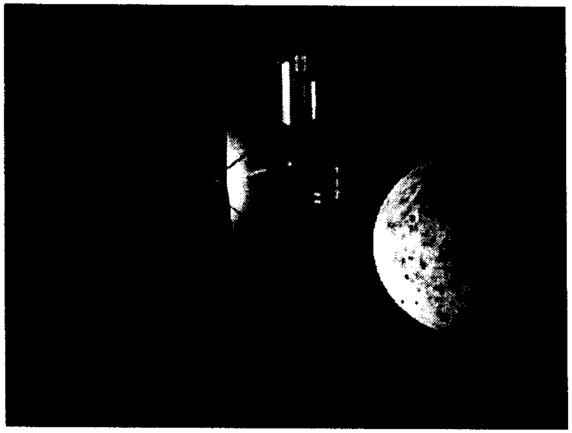


High Data Rate Instruments

Contact: Daniel S. Katz, JPL/Caltech, 4800 Oak Grove Drive, MS 168-522, Pasadena, California, 91109-8099, USA, Daniel.S.Katz@jpl.nasa.gov, phone: 818.354.7359, fax: 818.393.3134

REE Vision

Move Earth-based Scalable Supercomputing Technology into Space



Background

- Funded by Office of Space Science (Code S) as part of NASA's High Performance Computing and Communications Program
- Started in FY1996

REE Impact on NASA and DOD Missions by FY03

Faster - Fly State-of-the-Art Commercial Computing Technologies within 18 months of availability on the ground

Better - Onboard computer operating at > 300MOPS/watt scalable to mission requirements *(> 100x Mars Pathfinder power performance)*

Cheaper - No high cost radiation hardened processors or special purpose architectures

Objectives

- **High Power Performance:**
  - Obtain power efficiencies of 300-1000 MOPS per watt
  - Develop an architecture that scales to 100 watts (depending on mission needs)

Computational Testbeds

- **Fault-tolerant system software:**
  - Enable reliable operation for 10 years and more
  - Using commercially available or derived components

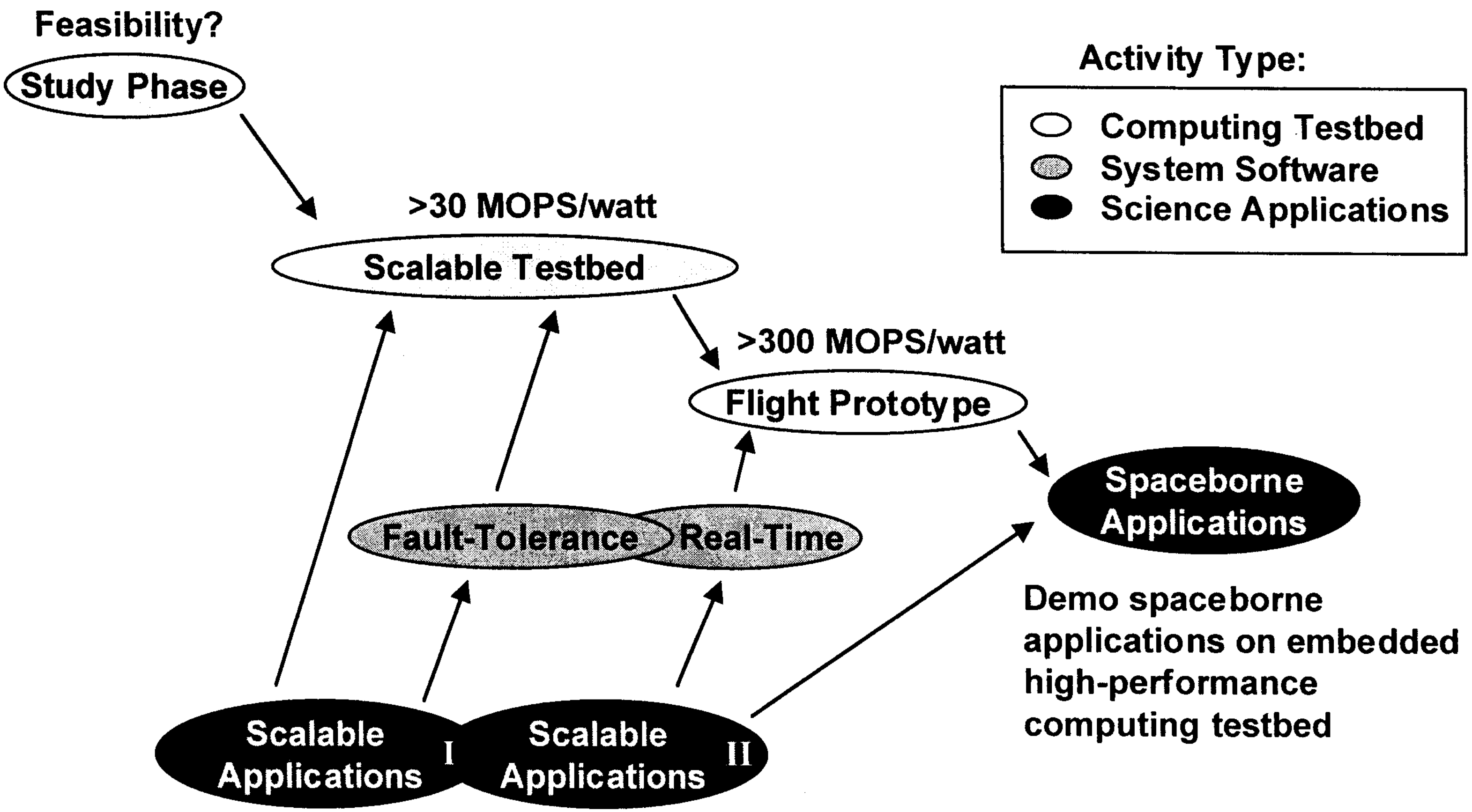
System Software

- **New spaceborne applications:**
  - Run in embedded high-performance computers
  - Return analysis results to the earth; not just raw data

Science Applications

Remote Exploration and Experimentation Project

Overview



REE Implementation

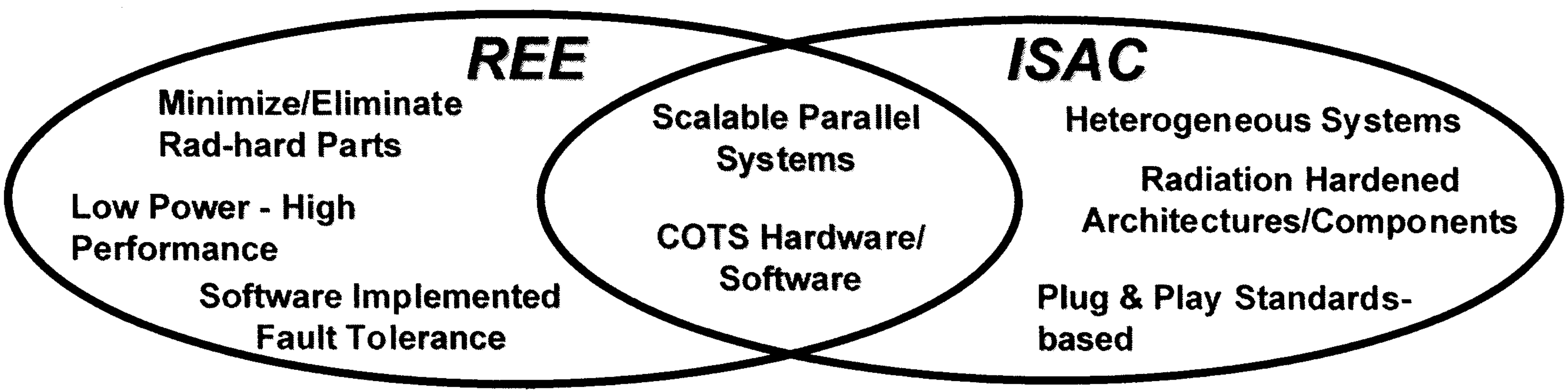
- Use COTS hardware and software to the maximum extent possible
  - Assume that memory supports EDAC
  - Assume hardware detection of "standard" exceptions, but assume that some faults will go undetected
  - Fault tolerance achieved through software
- Keep overhead low
  - Emphasize techniques which do not require replication
- Maintain architecture independence
  - Design should not be tied to any particular hardware architecture
- "95%" rule
  - System does not have to be continuously available
  - Reset is acceptable recovery technique
- Target large applications, both parallel and distributed
  - Gigabytes of memory, gigaflops of processing
  - Scalable with high efficiency
  - Static load balancing sufficient

Current Partnerships

USAF Phillips Lab

Improved Space Architecture Concepts (ISAC)

- Inter-program coordination on a regular basis
- Joint participation on technical reviews and procurement actions
- Technical interactions to avoid duplicate investments and identify possibilities for joint investment

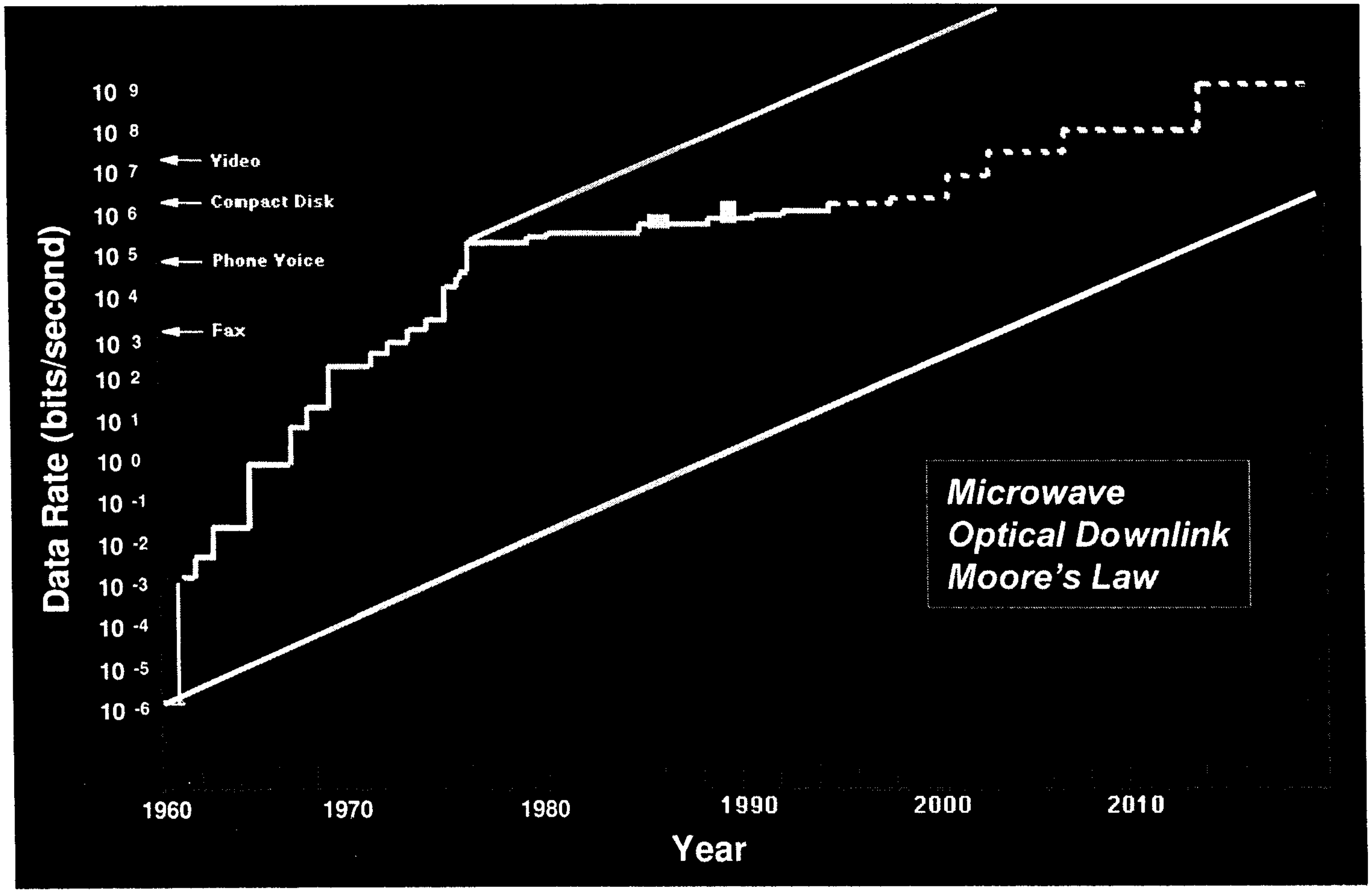


Science Application Teams

- Background
  - Enabling new and better science is a primary goal for REE
  - A new generation of Mission Scientists is emerging which sees the value of significant onboard computing capability
    - Mission Scientists still want the most data bits possible sent back to the ground
    - But bandwidth to the ground is stagnant, while instrument data rates continue to rise dramatically
    - Ground operations costs are a major component of mission costs
- Science Application Teams chosen to:
  - Represent the diversity of NASA onboard computing of the future
  - Drive architecture and system software requirements
  - Demonstrate the benefit of highly capable computing onboard
- Science Application Teams will:
  - Prototype applications based on their mission concepts
  - Port and demonstrate applications on the 1st Generation Testbed
  - Use their experiences with REE to influence some of their mission design decisions

Remote Exploration and Experimentation Project

Equivalent Downlink Bandwidth from Jupiter

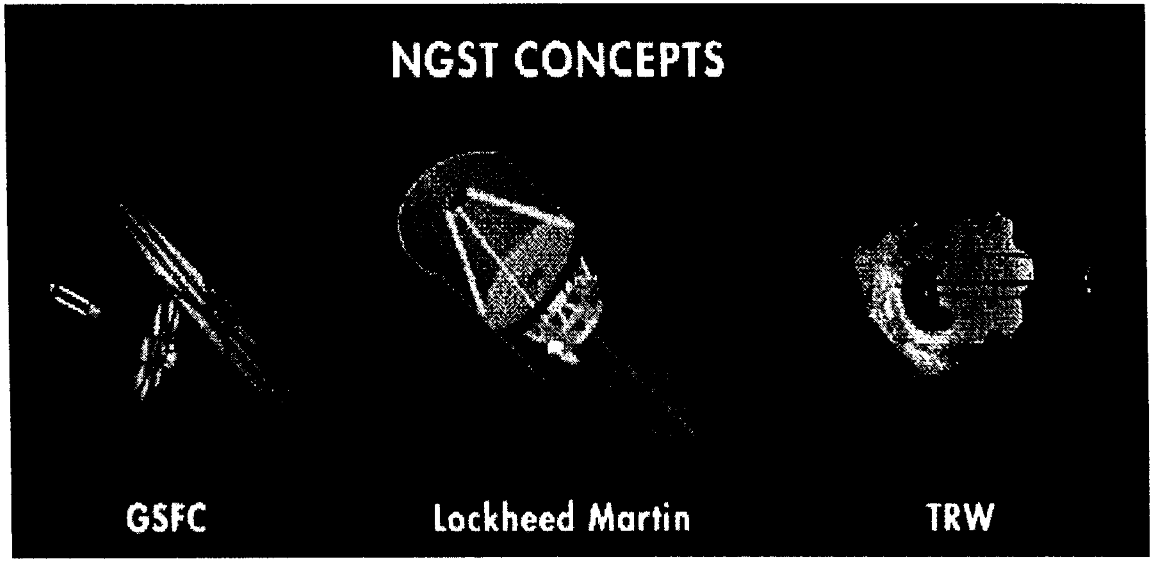


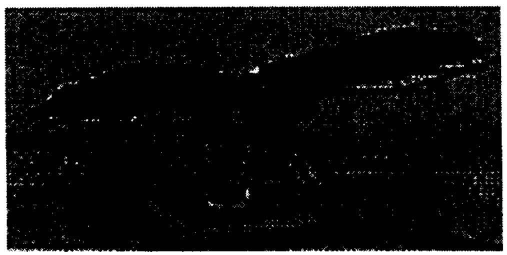
Remote Exploration and Experimentation Project

Next Generation Space Telescope Team

REE Principle Investigator: Dr. John Mather, NGST Study Scientist

SCIENCE OBJECTIVES
- Study the birth of the first galaxies
- Determine the shape and fate of the universe
- Study formation of stars and planets
- Observe the chemical evolution of the universe
- Probe the nature of dark matter

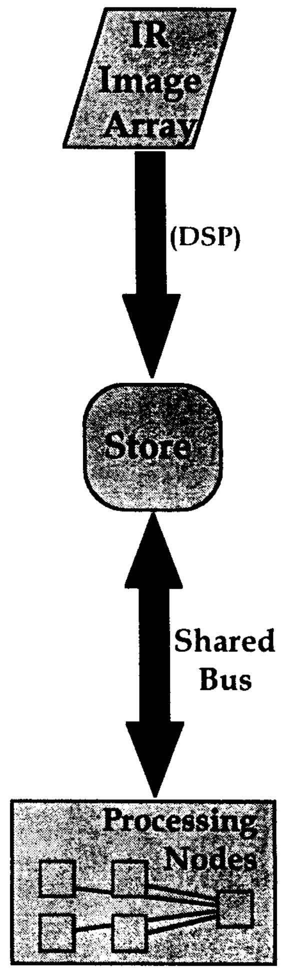




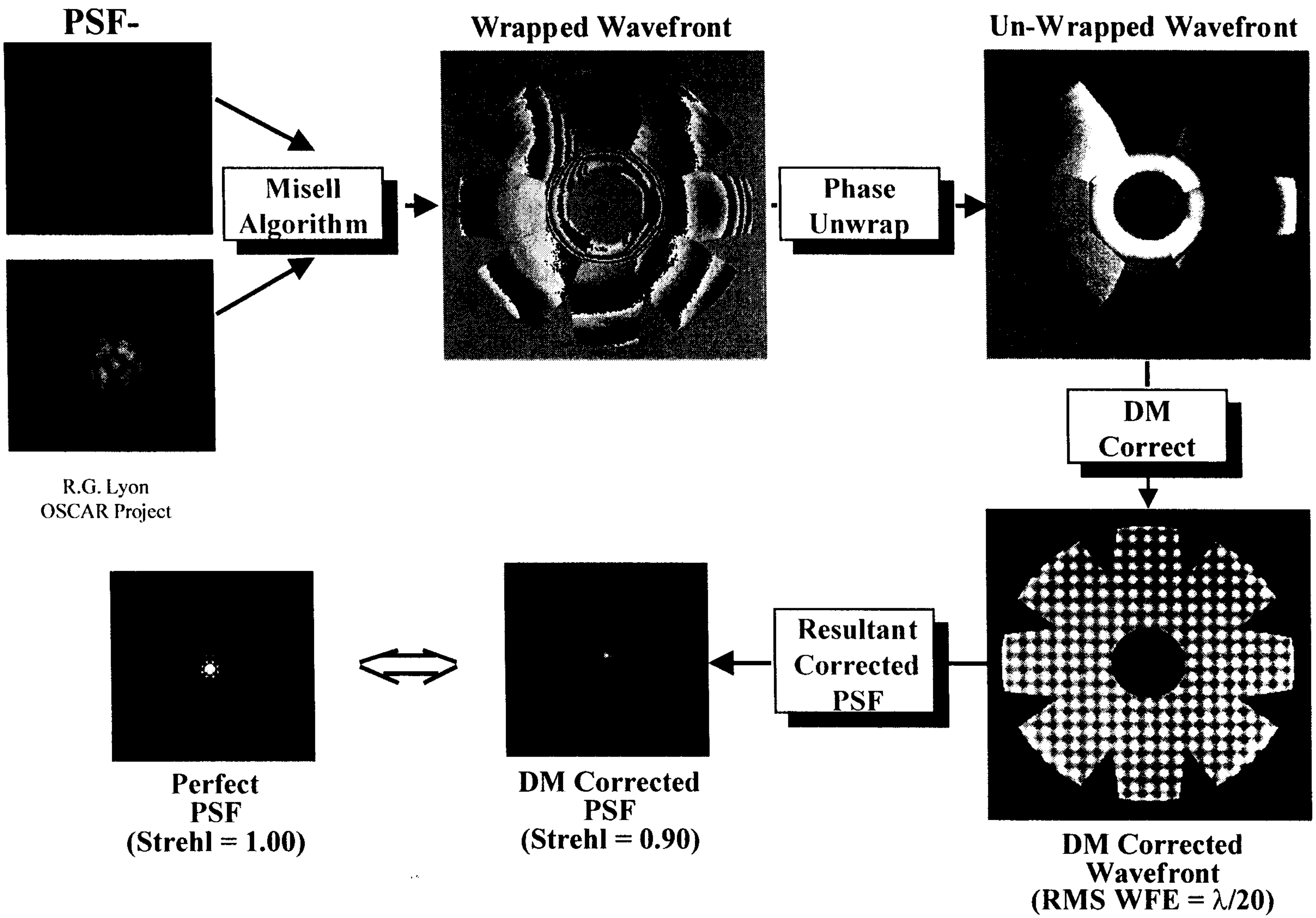
TECHNOLOGY HIGHLIGHTS
- Precision deployable and inflatable structures
- Large, low area density cold active optics
- Simulation based design
- Passive cooling
- Autonomous operations and onboard scheduling

NGST Hardware/Software Requirements

- General Configuration (tentative)
  - Sensing array feeds shared store through DSP glue
  - Image blocks (1Kx1K) stored in files and accessed by parallel nodes through shared bus (50 MB/s)
  - Highly data-parallel; little code parallelism desired
  - Many opportunities for data sanity checks, especially in optical calibration
- Image Processing
  - Fast scan of a large volume of image data to reject bad pixels
  - Image compression (possibility of feature identification)
  - Significant I/O per flop, but little IPC
- On-Board Optical Calibration
  - Reads image, extensive iteration, adjusts actuators
  - 2D FFT is iteration's core: low I/O per flop, but significant IPC



NGST Fine Figure Control Loop

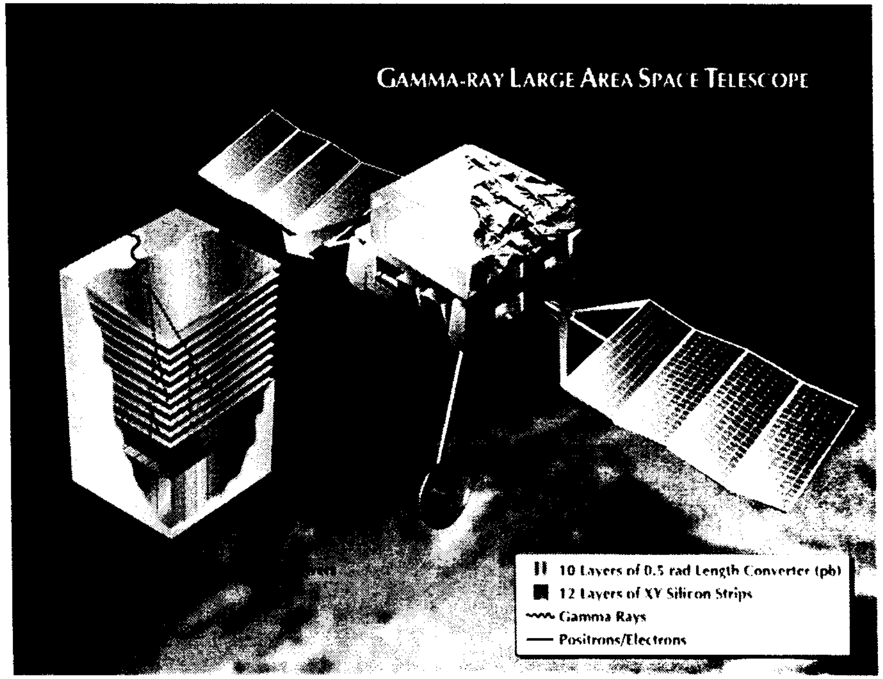


Remote Exploration and Experimentation Project

Gamma Ray Large Area Space Telescope

REEE Principal Investigator: Professor Peter Michelson, Stanford University, GLAST Principle Investigator

- GLAST will probe active galactic nuclei (spectral shape and cutoff), study gamma-ray pulsars, respond in real-time to gamma-ray bursts.
- GLAST will produce 5-10 Megabytes per second after sparse readout, mapping into 50 MIPS of computing requirements to meet the requirements for the baseline mission.
- New science addressed by GLAST focuses on transient events of a few days in AGNs and .01–100 seconds in gamma-ray bursts.
- REE could enable GLAST to produce 10x this data volume if it were to do most of its background discrimination in software. This would allow real-time identification of gamma-ray bursts, and permit the mission scientists to extract secondary science from the "background."



GLAST is a high-energy gamma-ray observatory designed for making observations of celestial sources in the range from 10 MeV to 300 GeV.

GLAST Triggering System

Trigger Criteria

Hardware:
Level I trigger causes strip detector states to be latched and read out to tower DRAM.

Level I — Three Consecutive Layers

3 kHz

Software: ~ 2 Kops/event
Same process runs on *each* tower using only data *local* to the tower.

Level II — Linearity and Anti-veto

A Level II trigger by *any* tower requires data to be assembled from all towers for Level III analysis.

900 Hz

Software: ~ 1 Mops/event
"Share" load over pool of processors.

Level III — Full Reconstruction

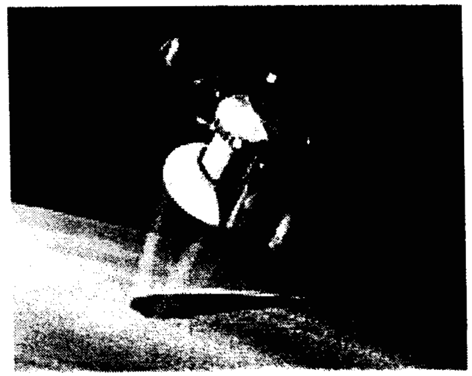
20 Hz

Cache until downlink opportunity

Orbiting Thermal Imaging Spectrometer

REE Principal Investigator - Alan Gillespie/U. Washington, Member of the ASTER Science Team

- Similar to Sacagawea:
  - Polar-orbiting high-resolution imaging infrared spectrometer (8-12 μm)
  - 64 bands of 12-bit data over a 21 swath at 30 m/pixel every 3.1 sec
  - Raw data rate of 30 MB/s
  - Designed to map emissivity of the Earth's surface to:
    - Map lithologic composition
    - Enable surface temperature recovery over all surfaces



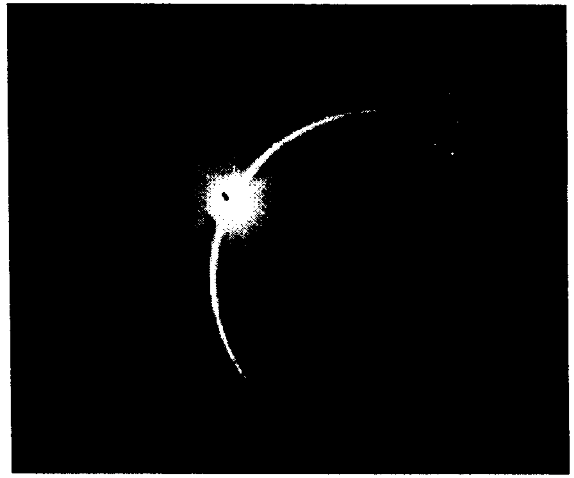
- Onboard Processing
  - Characterize and compensate for atmospheric effects
  - Calculate land surface temperatures and emissivity spectra
  - Automatically convert the emissivity data to a thematic map

Orbiting Thermal Imaging Spectrometer Data Flow

- Data Acquisition and Calibration
- Atmospheric Compensation
- Surface Temperature Estimation
- Atmospheric Analysis and Correction
- Temperature/Emissivity Separation
- Correction for Sky Irradiance
- Image Analysis
- Data Compression
- Transmit

Solar Terrestrial Probe Program

REE Principal Investigator - Steve Curtis/GSFC STPP Study Scientist



- Solar Terrestrial Probe Goal
  - Real-time quantitative understanding of the flow of energy,mass,momentum and radiation from the sun to the earth
    - Solar processes, flares and mass ejections
    - Interplanetary space and solar wind
    - Earth's magnetosphere and upper atmosphere
- Mission Onboard Processing Applications - Data Reduction!
  - Magnetospheric Constellation Mission
    - 50- 100 identical, spinning 10 kg spacecraft with on-board plasma analyzers (ions and electrons), a magnetometer and an electrometer
    - Compute moments of a sample plasma distribution function onboard
  - Low Frequency Radio Astronomy Imaging (ALFA/SIRA mission)
    - 16 - 64 formation flying spacecraft using interferometry to produce low frequency maps and two dimensional imaging of solar disturbances.
    - Compute pairs of time series (120+) to find the correlation maximum

Solar Terrestrial Probe Control Flows

Magnetospheric Constellation

1. Read data
2. Gain calibration
3. Calculate spacecraft potential
4. Calculate plasma moments
5. Compensate for unreliable data
6. Fit Gaussian
7. Calculate moments
8. Assess results
9. Store moment data

SIRA

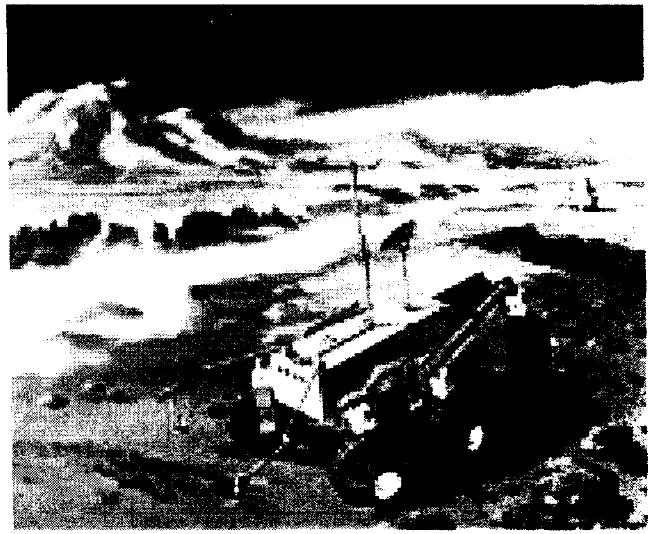
1. Read sensor data
2. Calculate Fourier transforms
3. Average spectrum in time
4. Filter and calibrate
5. Cross multiply to form spectra
6. Inverse Fourier transform of reduced data
7. Store Data

Autonomous Mars Rover Science

REE Principal Investigator: R. Steve Saunders/JPL Mars '01 Lander PI

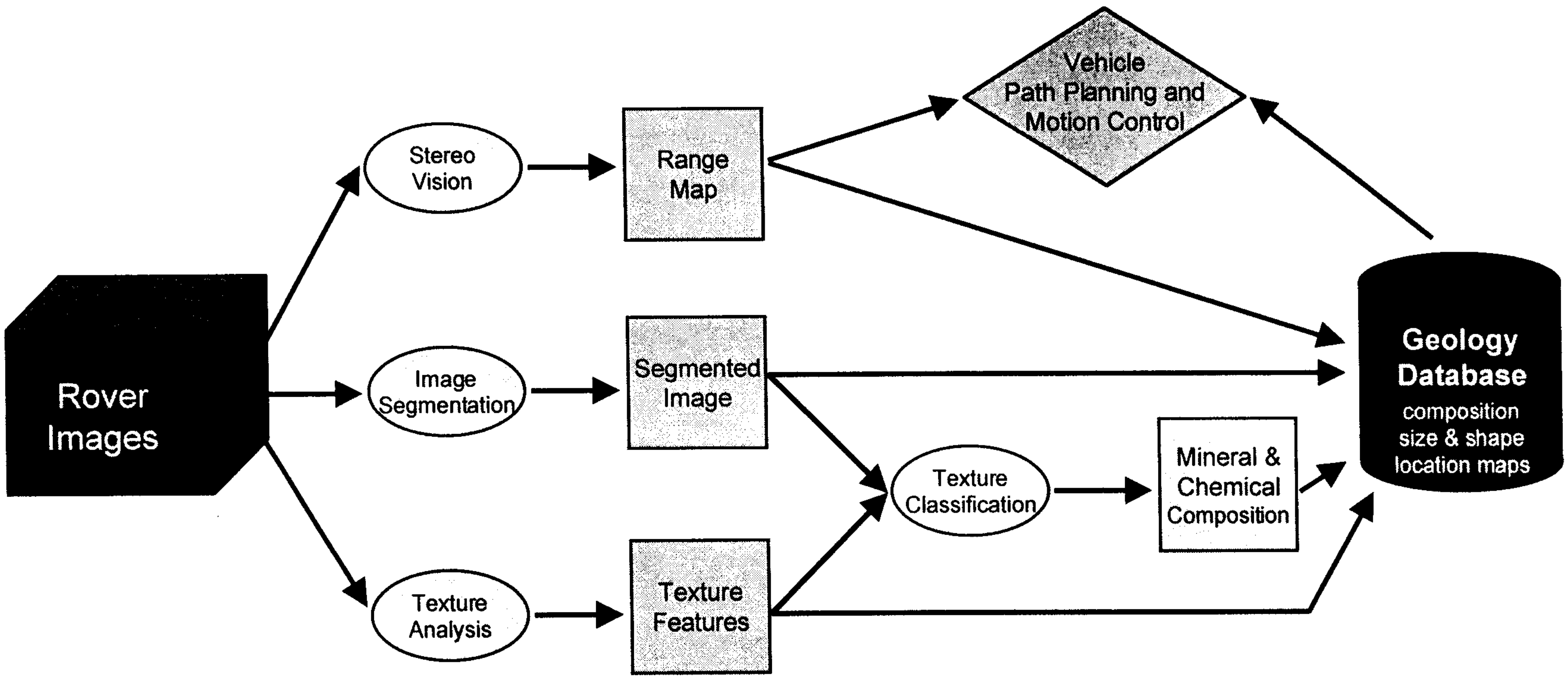
- Autonomous optimal terrain navigation
  - Stereo vision
  - Path planning from collected data
  - Autonomous determination of experiment schedule
  - Opportunistic scheduling

- Autonomous Field Geology
  - "Computational Geologist"
  - The rover returns analysis - not only data



Autonomous Mars Rover Science Application

Components for REE Testbed



Fault Tolerance

- Project Goals - High Performance with Low Power Using COTS
  - COTS will get us to high power performance
  - SEUs (radiation-induced Single Event Upsets) will be an issue
- Traditional Fault Tolerance Approaches for Spaceborne Systems
  - Radiation hardening
  - Replication
- Both approaches have a power performance penalty we can't live with!

Software Implemented Fault Tolerance

- Approach - Hardware/Software in Combination for a "95%" solution
  - Characterize the fault rates for "typical" (95% of) NASA missions
  - Characterize the range of application fault tolerance requirements
    - Simplex: Restart only for High Throughput Tasks
    - Duplex: Compare and restart only - for correct results which are not time critical
    - Triplex: Operate through
  - Partner with leading FT Experts to design "good enough" SIFT techniques
  - Validate SIFT techniques by testing and experimentation

- Remember - the missions which need REE most badly would, in our absence, have to throw away opportunities to acquire data!

Faults and Errors

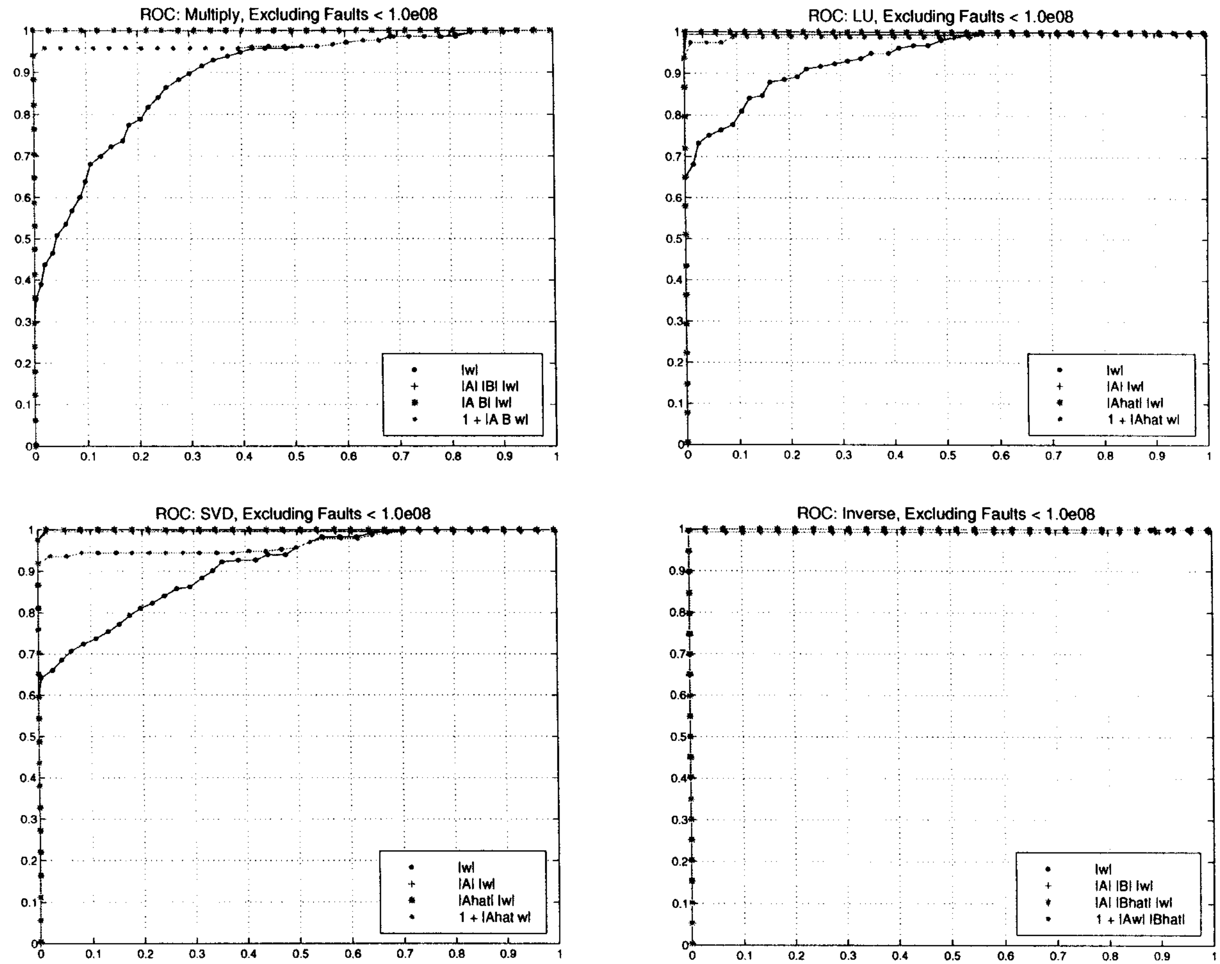
- Radiation environment causes faults
  - Most (>99.9%) of faults are transient, single event upsets (SEUs)
- Faults cause errors
  - Good Errors
    - Cause the node to crash
    - Cause the application to crash
    - Cause the application to hang
  - Bad Errors
    - Change application data
    - Application may complete, but the output may be wrong
- System Software can detect the good errors
  - Restarting the application/rollback/reboot is acceptable
- Applications must detect bad errors
  - Using Algorithm-Based Fault Tolerance (ABFT), assertion checking, other techniques

Algorithm-Based Fault Tolerance

- Started in 1984 with Huang and Abraham
  - Initial motivation was systolic arrays
  - Abraham and his students continued to develop ABFT throughout 1980s
- Relationship to convolutional coding noticed
- Picked up in early 90s by a group of linear algebraists (Boley et al., Boley and Luk)
- ABFT techniques exist for many numerical algorithms
  - Matrix multiply, LU decomposition, QR decomposition, single value decomposition (SVD), fast Fourier transform (FFT)
  - Require an error tolerance
    - setting of this error tolerance involves a trade-off between missing errors and false positives
- ABFT can correct as well as detect errors
  - Currently, we are focusing on error detection, using result checking

Remote Exploration and Experimentation Project

ABFT Results



Receiver Operating Characteristic (ROC) curves (fault-detection rate vs. false alarm rate) for random matrices of bounded condition number ($< 10^8$), excluding faults of relative size $< 10^{-8}$

ABFT Results (cont.)

- We have implemented a robust version of ScaLAPACK (on top of MPI) which detects errors using ABFT techniques
  - To the best of our knowledge, this is the first wrapping of a general purpose parallel library with an ABFT shell
  - Interface the same as standard ScaLAPACK with the addition of an extra error return code
  - For reasonable matrices, we can catch >99% (>97% for SVD) of significant errors with no false alarms
- ABFT version of FFTW recently completed, not yet fully tested
  - Interface the same as standard FFTW with the addition of an extra error return code

REE Results-to-Date

- Scalable applications have been delivered
  - 8 of 9 proposed applications have been delivered to JPL
  - 3 are currently running on an embedded system
- ABFT-wrapped libraries have been developed for linear algebra, FFT
  - Linear algebra routines have been rigorously tested
  - Next step is for the applications to use these libraries under fault injection experiments
- Similar progress is being made in the other REE activities
  - Zeroeth generation testbeds on-line at JPL
    - Beowulf cluster and prototype embedded system
  - First generation embedded testbed is being fabricated by Sanders
    - Delivery to JPL scheduled for 11/99
  - System software is being developed
    - Fault injector, fault detection and recovery mechanisms, scheduler, etc...
- A number of questions still need to be answered...

Open Questions

- What fault rates will occur?
  (radiation environment is known; affects of environment is unknown)
- What percentage of faults can be detected without replication?
  (using ABFT and other techniques to check for incorrect answers)
- What is the overhead and coverage of ABFT?
- Is checkpointing/rollback sufficient to recover from faults?
- Can the state of REE applications be made sufficiently small that the overhead of checkpointing is not prohibitive?